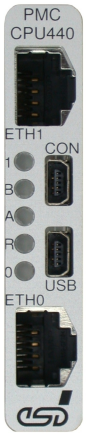
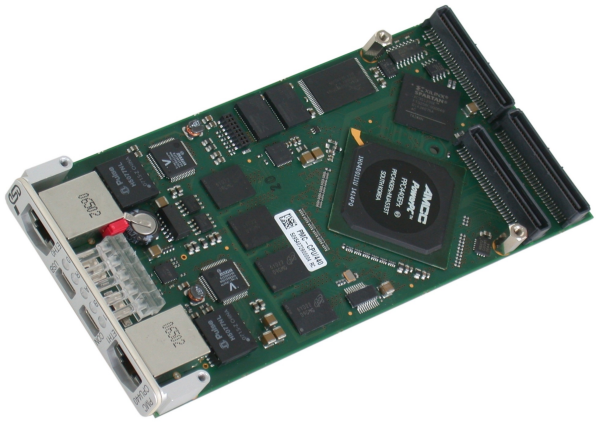
PMC-CPU/440

PowerPC 440EPx PrPMC module





User Manual

Product Order No. V.2027.02

Der Inhalt dieses Handbuches wurde mit größter Sorgfalt erarbeitet und geprüft.

esd übernimmt jedoch keine Verantwortung für Schäden, die aus Fehlern in der Dokumentation resultieren könnten. Insbesondere Beschreibungen und technische Daten sind keine zugesicherten Eigenschaften im rechtlichen Sinne.

esd hat das Recht, Änderungen am beschriebenen Produkt oder an der Dokumentation ohne vorherige Ankündigung vorzunehmen, wenn sie aus Gründen der Zuverlässigkeit oder Qualitätssicherung vorgenommen werden oder dem technischen Fortschritt dienen.

The information in this document has been carefully checked and is believed to be entirely reliable.

esd makes no warranty of any kind with regard to the material in this document, and assumes no responsibility for any errors that may appear in this document.

esd reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance or design.

esd assumes no responsibility for the use of any circuitry other than circuitry which is part of a product of **esd** gmbh.

esd does not convey to the purchaser of the product described herein any license under the patent rights of **esd** gmbh nor the rights of others.

---

esd electronic system design gmbh

Vahrenwalder Str. 207

D-30165 Hannover

GERMANY

Tel.: +49-511/372 98-0

FAX : +49-511/372 98-68

E-Mail: info@esd.eu

www.esd.eu

Dokumenten Information

document-no.:	V.2027.21
document type:	DOC08xx
document status:	preliminary
document revision:	1.1
date of creation:	2008-07-14
document path:	I:\Prj\esd\PMC\pmc440\Doku\
document filename:	pmc440-me.odt
number of pages / annex:	49 / 0

Responsible for content / author

Name	Company / Department	Phone	Email
M. Fuchs	esd gmbh / SD	+49 511 37298 0	support@esd-electronics.com

Distribution / Review

Name	Firma / Abteilung	Telefon	Email
-	-	-	-

Modification history

Technical modifications in this overview are marked by an additional "**!**".

The following overview is handled from revision 1.0.

!	Revision	Chapter	Page	Changelog	Date / Sign
	0.9	all	all	Document created	2008-03-18 / MF
	1.0	all	all	Initial release	2008-03-28 / MF
		11.5.3	25	Added 'sbe' command describtion	2008-04-01 / MF
		9.2	20	Added chapter	2008-04-09 / MF
		11.5.5	26	Chapter rewritten	
		14	46	Added Appendix (GPL License)	2008-05-09 / MF
		12.2 12.6	31	Chapters added	2008-07-04 / MF

Contents

1 Overview

1.1 Description Of The PMC-CPU/440 Module

The PMC-CPU/440 is a PMC module in ‘single’ PCI Mezzanine Card form factor. It can act as PMC monarch (PrPMC) or as non-monarch (adapter/target) board. Apart from a powerful CPU core the PowerPC 440EPx embedded processor integrates a DDR2 RAM controller, a PCI bus interface, a controller for serial interfaces and two gigabit ethernet MACs.

The module comes with DDR2 RAM and flash memories, a double layer capacitor buffered realtime clock (RTC) and a FPGA. A serial console is provided through an USB device port that integrates a USB-to-serial converter. This is in accordance to modern PC technologies abandoning serial ports. A second serial interface is accessible via the PMC-I/O connector with RS232 signal levels.

The module provides two 1000 BaseT gigabit ethernet port on the front panel. Their link status and several other status information are indicated by 5 LEDs.

The on-board FPGA provides extended flexibilities for specialized applications. The default FPGA functionality includes two high speed CAN interfaces, IRIG-B timecode decoding and generation and much more. Many FPGA pins are directly connected the PMC P4 IO connector.

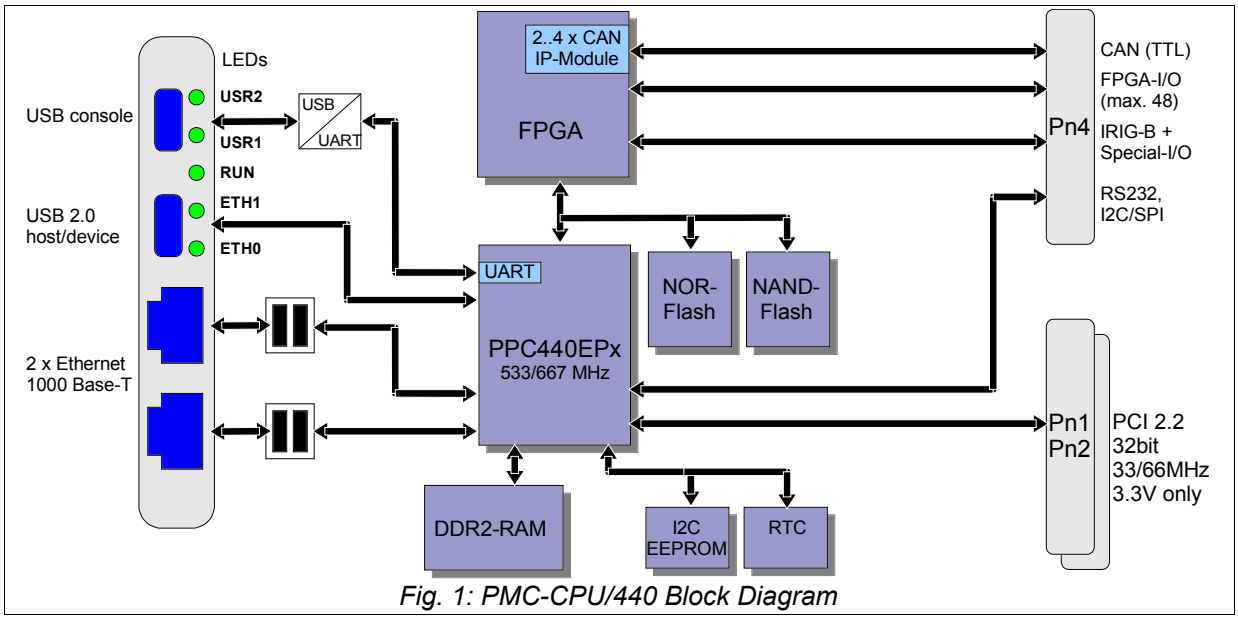


Fig. 1: PMC-CPU/440 Block Diagram

1.2 Technical Data

1.2.1 General

Operating temperature	0...50 °C
Humidity	max. 90%, non-condensing
Power supply	5V and 3.3V DC, ±5%
Power consumption	~ 6 W (since hardware revision 1.2 main powerrail is 3.3V, before main powerrail is 5.0V)
PCB formfactor	148mm x 74mm
weight	~ 140g (including heatsink)

1.2.2 CPU Core

CPU	PPC440EPx, AMCC
CPU clock	533MHz (optional: 667MHz)
RAM	256MB, DDR2-RAM
Flash memory (NOR)	4MB (Spansion, S29AL032D90TFI04)
NAND flash	256 MB, (2KB page size)
EEPROM	a) 4KB connected via I2C (used for bootloader configuration) b) 256 Bytes connected via I2C (used for CPU strapping)
Watchdog	CPU internal, 4 selectable intervals: 4ms, 64ms, 1s, 16s (@533 MHZ CPU clock)

1.2.3 Realtime Clock (RTC)

Type	RX-8025, Epson
NVRAM	None
Backup energy source	Double layer capacitor
Backup time	Up to 1 week

1.2.4 PCI Interface

Specification	PCI 2.2 compatible
Features	- host or target - bus master - 2 target address spaces (BARs)
PCI clock	33/66 MHz
Bus width	32 bit
IO voltage	3.3V only (the PCI interface is **NOT** 5V tolerant)
Interrupt	non-monarch: INTA# monarch: INTA#..INTD#

1.2.5 Serial Interfaces

Number	2
Controller	CPU internal, 16550 compatible
Physical Layer	Port0: USB 1.1 device via FT232RQ[1] USB-to-serial converter Port1: RS232
Lines	RxD, TxD, CTS, RTS
Connector	Port0: USB Mini-B Port1: PMC P4
Baudrate	Max. 115200 baud Default baudrate on USB console is 115200 baud, 8N1

1.2.6 CAN Interfaces

Number	2
CAN controller	esd CAN IP core in FPGA
CAN protocol	CAN 2.0A/2.0B
Physical interface	TTL, no transceiver on-board
Bitrate	10 kbit/s ... 1 Mbit/s
Bus termination	None
Connectors	PMC P4

1.2.7 Ethernet Interface

Number	2
Standard	IEEE 802.3, 10/100/1000BaseT
Bitrate	10/100/1000Mbit/s
Controller	CPU internal
Isolation	transformer
Connector	RJ45 socket in frontpanel
MAC-address	Port0: 00:02:27:83:40:00 + (serial# – 1) * 2 Port1: 00:02:27:83:40:00 + (serial# – 1) * 2 + 1

1.2.8 USB Interface (Host or Device)

Number	1
Standard	USB 2.0
Bitrate	480 Mbit/s
Controller	CPU internal, host (OHCI/EHCI) alt. device
Role	Host or device, configurable via software
Connector	Mini-AB in frontpanel

1 esd provides a suitable software driver for Linux or MS Windows PCs.

2 Installation Instructions

- The PMC-CPU/440 comes with a separate frontpanel sealing (rubber band) and four M2.5 x 5mm mounting screws. Use only these M2.5 x 5 screws.
- ***Attention:*** Do not use longer screws because they might damage the PMC-CPU/440 mounting threads.
- The PMC-CPU/440 may only be used on PMC sites with 3.3V PCI signalling. Typically this is ensured by the correct IO-voltage coding holes in the PMC module and pins on the carrier system.

3 Frontpanel

The PMC-CPU/440 frontpanel provides access to the Ethernet interfaces, USB console (CON) and USB 2.0 host/device port. Five LEDs indicate several status information:

LED designator	Color	Function
LED 0	green	Ethernet port 0 link/traffic indicator. This LED is lit when the ethernet port has established a link. Flickering of this LED indicates network dataflow.
LED 1	green	Ethernet port 1 link/traffic indicator. This LED is lit when the ethernet port has established a link. Flickering of this LED indicates network dataflow.
LED A	green	User LED 'A'. This LED can be controlled by software. When using the FPGA internal CAN IP cores this LED can be configured to indicate CAN data traffic.
LED B	green	User LED 'B'. This LED can be controlled by software. When using the FPGA internal CAN IP cores this LED can be configured to indicate CAN data traffic.
LED R	red / green	This LED will lit red after power on until the bootloader has finished hardware intialisation. It can be turned on in green under software control.

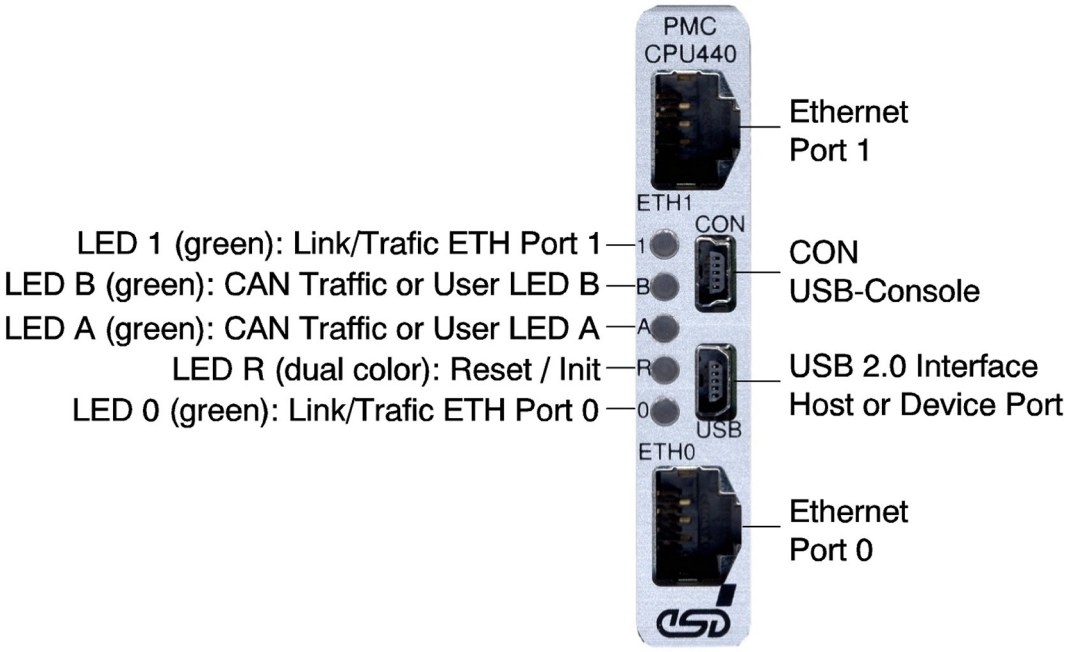


Fig. 2: PMC-CPU/440 Frontpanel

4 Top and Bottom Side

The top side of the PMC-CPU/440 PCB is covered by a an aluminium heatsink on most areas. There are two 0,1“ contacts near the frontpanel. These can be used to install a *strapping jumper*. By default the jumper is not installed when the boards are shipped. In this case the CPU clocking and some other parameters are configured through the content of a serial EEPROM. With the jumper installed a default configuration is active.

Jumper designator	Function
JP2	***Installed:*** Default configuration: CPU runs at 533 Mhz and serial bootloader console is on serial port 1 on PMC P4 connector. ***Not installed:*** Configuration is read from I2C EEPROM. The EEPROM content can be configured by the *sbe* bootloader command.

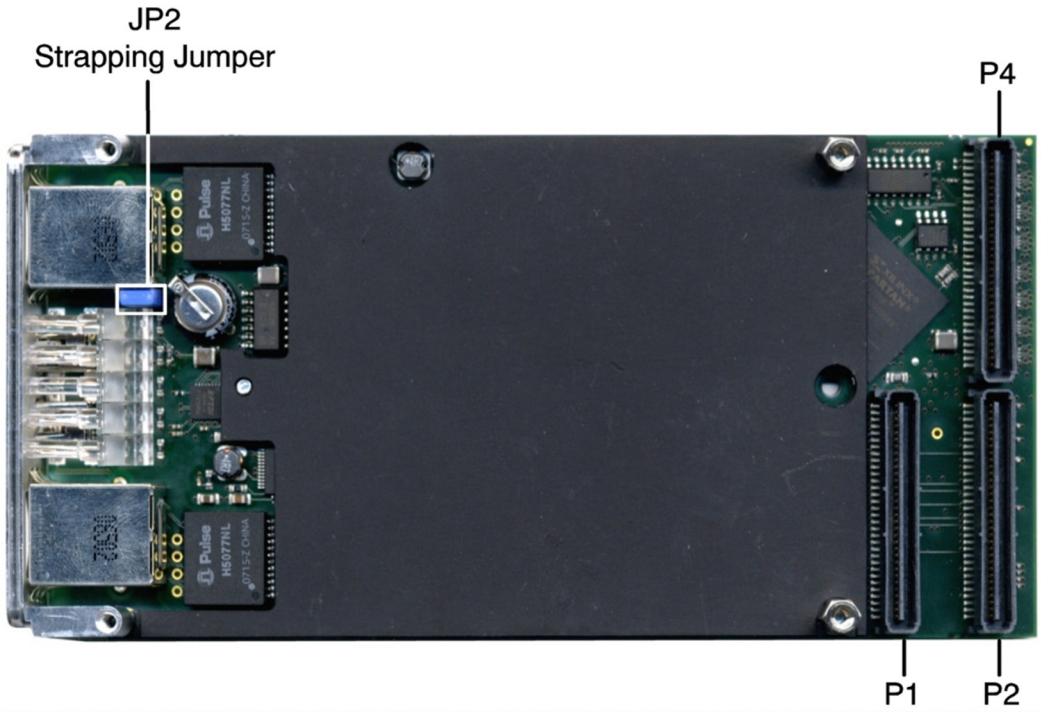


Fig. 3: Topview of the PMC module

The PMC-CPU/440 has one additional LED on the bottom side of the PCB. This LEDs is visible when the PMC module is installed on a carrier board.

LED designator	Color	Function
LED6	green	Data indicator for USB console (CON).

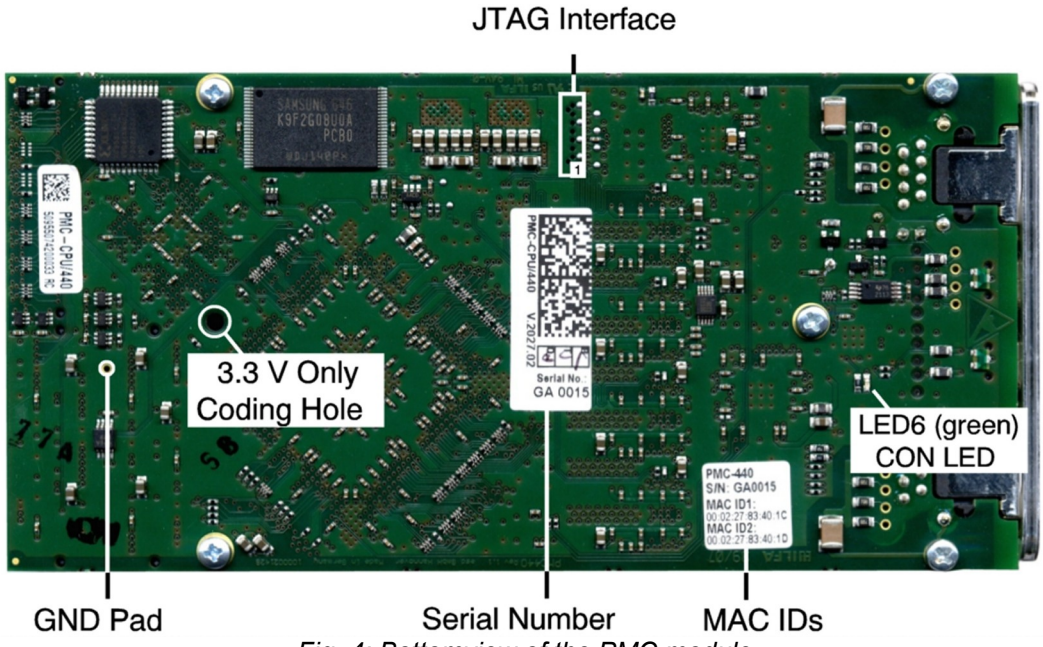


Fig. 4: Bottomview of the PMC module

5 PPC440EPx GPIO Functions

This chapter will be added later!

6 JTAG Debug Interface

The debug port is used during manufacturing tests and to flash an initial firmware.

6.1 JTAG Chain Description

All JTAG capable circuits on the PMC440 module are connected to a common JTAG chain in the order as follows:

TDI -> **CPU** -> **FPGA** -> **CPLD -> Phy0 -> Phy1** -> TDO

Device position in chain	Function	Device part number	JTAG IR length
1	CPU	PPC440EPx (AMCC)	8
2	FPGA	XC2S1200E-FT256 (Xilinx)	6
3	CPLD	XC9536XL-VQ44 (Xilinx)	8
4 + 5	Ethernet Phy	VCS8601 (Vitesse)	2 x 4

6.2 JTAG Connector (X5)

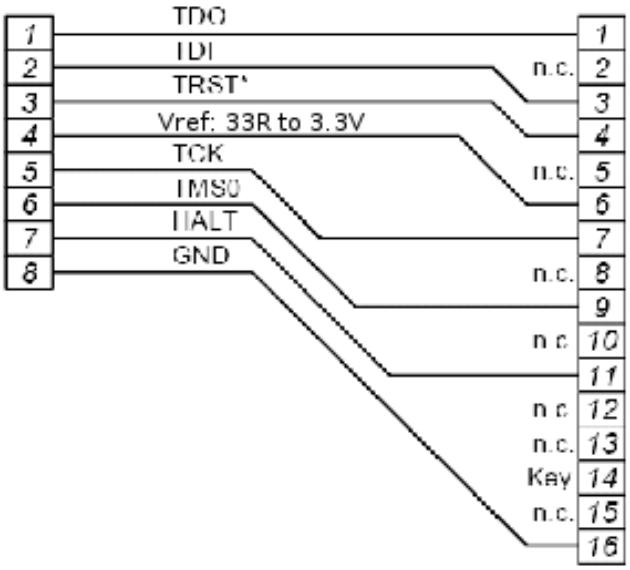
The JTAG interface can be accessed through a 8-pin single in-line 1,27mm plug. It is possible to connect to the JTAG interface even when the PMC440 module is assembled on a carrier system. Drillings in the PCB allow interfacing the JTAG port from the solder side of the module. It can be connected via a contact strip connector. It is recommended to build a simple adapter from the contact strip connector to a 16-pin connector to connect to the port. This 16-pin connector is used by a wide range of third party hardware debugger vendors.

Attention: Be careful when plugging the contact strip connector into the drillings. Do not plug it in to deep. The pins must not come into contact with the module's heatsink.

Pin Number X5	Function	Direction
1[2]	TDO	output
2	TDI	input
3	TRST# (pulled down via 1kOhm)	input
4	Vref+ (3.3V via 33 Ohm resistor)	output
5	TCK	input
6	TMS	input

2 The JTAG connector is oriented with pin 1 close to the center of the PCB. The pin 1 drilling is marked by a tiny '1' on the PCB's solder side.

Pin Number X5	Function	Direction
7	PPC440 HALT	input
8	GND	reference



Example of an Adapter (Uncased)

Connectors to build adapter:

SMD-pin contact strip: Fa. Samtec, 'modified pin contact strip', order-no. MTMS-108-58-T-S-485

2.54 mm post connector: i.e. Fa. Harting, 16-pole, straight, order-no. 09185167324

Fig. 5: self-build JTAG adapter inserted into PMC module from botton side of PCB

7 PMC-Connectors

The PMC-CPU/440 module uses the PMC connectors P1, P2 and P4. P1 and P2 provide the PCI interface and power supply connection. P4 has a complete module specific pinout.

7.1 PMC P1 Connector

Pin	Signal	Signal	Pin
1	n.c. (TCK)	-12V	2
3	GND	INTA#	4
5	INTB#	INTC#	6
7	GND (PRESENT#)	+5V	8
9	INTD#	n.c. (reserved)	10
11	GND	n.c. (reserved)	12
13	PCI-CLK	GND	14
15	GND	GNT#	16
17	REQ#	+5V	18
19	VIO	AD[31]	20
21	AD[28]	AD[27]	22
23	AD[25]	GND	24
25	GND	C/BE3#	26
27	AD[22]	AD[21]	28
29	AD[19]	+5V	30
31	VIO	AD[17]	32
33	FRAME#	GND	34
35	GND	IRDY#	36
37	DEVSEL#	+5V	38
39	GND	n.c. (LOCK#)	40
41	n.c. (SDONE#)	n.c. (SBO)	42
43	PAR	GND	44
45	VIO	AD[15]	46
47	AD[12]	AD[11]	48
49	AD[09]	+5V	50
51	GND	C/BE0#	52
53	AD[06]	AD[05]	54
55	AD[04]	GND	56
57	VIO	AD[03]	58
59	AD[02]	AD[01]	60
61	AD[00]	+5V	62

63	GND	n.c. (REQ64#)	64

7.2 PMC P2 Connector

Pin	Signal	Signal	Pin
1	+12V	n.c.	2
3	n.c.	TDO (bridged to TDI)	4
5	TDI (bridged to TDO)	GND	6
7	GND	n.c. (reserved)	8
9	n.c. (reserved)	n.c. (reserved)	10
11	MODE2#	+3.3V	12
13	PCI-RST#	MODE3#	14
15	+3.3V	MODE4#	16
17	n.c. (PME#)	GND	18
19	AD[30]	AD[29]	20
21	GND	AD[26]	22
23	AD[24]	+3.3V	24
25	IDSEL	AD[23]	26
27	+3.3V	AD[20]	28
29	AD[18]	GND	30
31	AD[16]	C/BE2#	32
33	GND	IDSELB	34
35	TRDY#	+3.3V	36
37	GND	STOP#	38
39	PERR#	GND	40
41	+3.3V	SERR#	42
43	C/BE1#	GND	44
45	AD[14]	AD[13]	46
47	M66EN	AD[10]	48
49	AD[08]	+3.3V	50
51	AD[07]	n.c. (REQB#)	52
53	+3.3V	GNTB#	54
55	n.c. (reserved)	GND	56
57	n.c. (reserved)	EREADY	58
59	GND	RESETOUT#	60
61	n.c. (ACK64#)	+3.3V	62
63	GND	MONARCH#	64

7.3 PMC P4 I/O Connector

P4 is used to interface many PMC-CPU/440 specific interfaces like the CAN buses, I2C and FPGA-I/Os. esd offers a PMC-PIM module with two isolated CAN physical circuits.

7.3.1 Pinout

Pin	Signal Name	Notes	Signal Name	Notes	Pin
1	FPGA-IO<0>	3.3V, IO	FPGA-IO<1>	3.3V, IO	2
3	FPGA-IO<2>	3.3V, IO	FPGA-IO<3>	3.3V, IO	4
5	FPGA-IO<4>	3.3V, IO	FPGA-IO<5>	3.3V, IO	6
7-27	*FPGA-IO<6..26>*	*3.3V, IO*	*FPGA-IO<7..27>*	*3.3V, IO*	*8-28*
29	FPGA-IO<28>	3.3V, IO	FPGA-IO<29>	3.3V, IO	30
31	FPGA-IO<30>	3.3V, IO	FPGA-IO<31>	3.3V, IO	32
33	+5V (FPGA-IO<32>)	PWR, 5V	CAN0_TX (FPGA-IO<33>)	5V, O	34
35	n.c. (FPGA-IO<34>)	(3.3V, IO)	CAN0_RX (FPGA-IO<35>)	5V, I	36
37	n.c. (FPGA-IO<36>)	(3.3V, IO)	CAN1_TX (FPGA-IO<37>)	5V, O	38
39	n.c. (FPGA-IO<38>)	(3.3V, IO)	CAN1_RX (FPGA-IO<39>)	5V, I	40
41	n.c. (FPGA-IO<40>)	(3.3V, IO)	GND (FPGA-IO<41>)	PWR, GND	42
43	GND (FPGA-IO<42>)	PWR, GND	n.c. (FPGA-IO<43>)	(3.3V, IO)	44
45	n.c. (FPGA-IO<44>)	(3.3V, IO)	RxS1	RS232, I	46
47	RTSS1	RS232, O	TxS1	RS232, O	48
49	CTSS1	RS232, I	n.c. (FPGA-IO<45>)	(3.3V, IO)	50
51	n.c. (FPGA-IO<46>)	(3.3V, IO)	GND (FPGA-IO<47>)	PWR, GND	52
53	CLOCK_IN	5V, I	CLOCK_OUT	3.3V, O	54
55	RESET_IN	5V, I	RESET_OUT	3.3V, O	56
57	IRIG-B_R_IN	5V, I	IRIG-B_R_OUT	3.3V, O	58
59	IRIG-B_R_P	RS485+	CLOCK_EN	3.3V, O	60
61	IRIG-B_R_M	RS485-	RESET_EN	3.3V, O	62
63	IIC1-SDA	3.3V, IO	IIC1-SCL	3.3V, IO	64

Notes:

1) Signal names and corresponding notes in braces are optional and only available in non-standard configurations.
2) The PMC440-NGCC variant uses the optional FPGA-IO signals: FPGA-IO<32..47>.
3) FPGA-IO<33,35,37,39> have dedicated signal directions!
4) All FPGA-IO signals have 22 Ohm series resistors.
5) Above notes have the folloing meaning:
   - 3.3V: 3.3V-only digital signal. These signals are not 5V tolerant.

- 5V: 5V tolerant input or 5V driving output.
- I: input only, O: output only, IO: bidirectional signal
- PWR: power supply signal (+5V, GND)
- n.c.: these signals are not connected on the PMC module. Special configurations may exists where these signals are used. Do not connect to these signals!
- RS232: RS232 signals (logic-1: -9V, logic-0: +9V)
- RS485+/-: RS485 differential signals

7.3.2 Signal Description

Signal Name	Direction	Description
CLOCK_IN	IN	3,3V/5V tolerant general purpose input. This signal is logically connected to a FPGA pin. CLOCK_IN can optionally be configured as a timestamp reference clock source.
CLOCK_OUT	OUT	LVTTL general purpose output. This signal is logically connected to a FPGA pin. CLOCK_OUT can optionally be configured as a timestamp reference clock output.
CLOCK_EN	OUT	LVTTL general purpose output. This signal is logically connected to a FPGA pin. CLOCK_EN can be used to enable an RS485 driver on a PMC-PIM module (application specific).
RESET_IN	IN	3,3V/5V tolerant general purpose input. This signal is logically connected to a FPGA pin. This pin can optionally be enabled to reset the FPGA internal timestamp counter.
RESET_OUT	OUT	LVTTL general purpose output. This signal is logically connected to a FPGA pin. This pin can be used to generate a defined reset pulse to other CLOCK/RESET sinks.
RESET_EN	OUT	LVTTL general purpose output. This signal is logically connected to a FPGA pin. RESET_EN can be used to enable an RS485 driver on a PMC-PIM module (application specific).
IRIG-B_R_IN	IN	3,3V/5V tolerant general purpose input. This signal is logically connected to a FPGA pin. This signal is used as IRIG-B time signal input when a TTL time signal is provided. This signal is logically connected to a FPGA pin.
IRIG-B_R_OUT	OUT	LVTTL general purpose output. This signal is reserved for future implementation of an IRIG-B time code generator.
IRIG-B_R_P	diff. I/O+	Half duplex RS485 differential IRIG-B input/output. This signal pair is used to supply the PMC440 with an external IRIG-B time signal source. This signal pair can also be configured as an output that is controlled by FPGA internal functionality (e.g. IRIG-B timecode generation).
IRIG-B_R_M	diff. I/O-	
TX0-C0#	OUT	TTL, CAN0 transmit
RX0-C0#	IN	TTL, CAN0 receive
TX0-C1#	OUT	TTL, CAN1 transmit
RX0-C1#	IN	TTL, CAN1 receive
RxS1	IN	RS232 receive data, 2nd port
TxS1	OUT	RS232 transmit data, 2nd port
RTSS1	OUT	RS232 request to send, 2nd port

Signal Name	Direction	Description
CTSS1	IN	RS232 clear to send, 2nd port
SDA_R	I/O	I2C bus, pulled-up against 3,3V
SCL_R	I/O	
GND	GND	Ground

Note: All output signals except the CAN signals are using 3.3V signalling. All inputs, but the I2C signals are 5V tolerant.

8 Local Memory Map

The PPC440EPx CPU uses more than 32 bits for physical addresses. The U-Boot bootloader configures the MMU in a way that physical addesses above 4GB will be mapped below the 4 GB border. The address translation just masks all high bits. Example: GPIO controller 0 has physical address 0x1ef600b00. It is mapped to 0xef600b00 by U-Boot. Please note that this may be handled differently depending on the used operating system.

Start-Address	End-Address	Function
0x0.0000.0000	0x0.0fff.ffff	SDRAM (256MB)
0x1.8000.0000	0x1.bfff.ffff	1GB PCI memory address space 0x8000.0000 – 0xbfff.ffff *Note:* This is the default setup as initialized by the U-Boot bootloader using the PMM0 register set. It can be changed during runtime by the operating system.
0x1.d000.0000	0x1.d00f.ffff	EBC[3] bank 2 (PerCS2#): NAND flash
0x1.ef00.0000	0x1.ef0f.ffff	EBC bank 4 (PerCS4#): FPGA (32 bit bank, size=1MB)
0x1.ef00.0000	0x1.ef00.00ff	FPGA internal registers (32bit bank)
0x1.ef01.n000	0x1.ef01.nfff	esd's CAN IP cores (default: 2 IP cores; n=0..1)
0x1.ef08.0000	0x1.ef0f.ffff	FPGA internal registers (reserved for custom extensions)
0x1.ef10.0000	0x1.ef1f.ffff	EBC bank 5 (PerCS5#): FPGA (16 bit bank, size=1MB)
0x1.ef18.0000	0x1.ef1f.ffff	FPGA internal registers (16 bit address space) (reserved for custom extensions)
0x1.ef60.0b00	0x1.ef60.0b1f	GPIO controller 0
0x1.ef60.0c00	0x1.ef60.0c1f	GPIO controller 1
0x1.ffc0.0000 0x1.fff0.0000 0x1.fff8.0000 0x1.fffa.0000	0x1.ffef.ffff 0x1.fff7.ffff 0x1.fff9.ffff 0x1.ffff.ffff	EBC bank 0 (PerCS0#): 4MB NOR flash - unused - default FPGA bit file image (512KB) - reserved (128KB) - U-Boot bootloader (384KB)

3 EBC = external bus controller

9 Interrupts

9.1 External Interrupt Assignment

UIC / IRQ#	Ext. Interrupt Pin	Device	Configurations
1 / 28	0	FPGA (IRQ0)	active low, level sensitive
1 / 30	1	FPGA (IRQ1)	active low, level sensitive
2 / 3	2	FPGA (IRQ2)	active low, level sensitive
2 / 4	3	Ethernet PHY 0	active low, level sensitive
0 / 27	4	Ethernet PHY 1	active low, level sensitive
2 / 0	5	RTC	active low, level sensitive
2 / 1	6	PCI (INTA#)	active low, level sensitive
1 / 18	7	PCI (INTB#)	active low, level sensitive
1 / 19	8	PCI (INTC#)	active low, level sensitive
1 / 20	9	PCI (INTD#)	active low, level sensitive

9.2 PCI Interrupt Handling

9.2.1 Monarch Mode

In monarch mode the PPC440EPx's external interrupts 6 to 9 (see table above) are used as PCI-INTA#..PCI-INTD# inputs.

9.2.2 Asserting PCI Interrupts In Non-monarch Mode

- Method I: set API flag in PCIC0_ICS register (see PPC440EPx usermanual)
- Method II:an alternative way to assert the PCI-INTA# line (as on the PMC-CPU/405) is not supported.

9.2.3 Asserting Local Interrupts From PCI Bus

- Method I: PCI master writes to PCI_COMMAND register.
- Method II:PCI master writes to the FPGA's HOSTCTRL reigster (see chap. 12.2)

10 PCI Configuration

The PMC-CPU/405 uses the following PCI identification:

Monarch (PrPMC)	Non-Monarch
Class/Subclasscode: 0x0600 (hostbridge) Vendor-ID: 0x1014 Device-ID:	Class/Subclasscode: 0x0b20 (processor, PPC) Vendor-ID: 0x1014 Device-ID:

Monarch (PrPMC)	Non-Monarch
0x027f Subsystem-Vendor-ID: 0x12fe (esd gmbh) Subsystem-ID: 0x0440	0x027f Subsystem-Vendor-ID: 0x12fe (esd gmbh) Subsystem-ID: 0x0441

PCI base address register mapping:

- PCI-BAR1:
  - default: 64MB; top of local SDRAM (0x0c00.0000-0x0fff.ffff)
  - mapping can be modified through bootloader environment variables. A minimum size of 4MB is recommended to allow firmware download to non-monarch boards via PCI. The local base can be reconfigured during runtime.
  - BAR1 configuration must not be changed when any esd drivers (PciAccess, backplane-network-driver) are used.
- PCI-BAR2:
  - size: 16MB
  - mapping of local address space 0x1.ef00.0000 – 0x1.efff.ffff
    - This means that the FPGA internal register (e.g. IRIG-B time) start at the beginning of BAR2.
  - offset 0x600b00/0x600c00: 440's GPIO controller (used for *remote self-reset#* and *adapter interrupt control* functions)
  - offset 0x000000: FPGA internal registers (IRIG-B time, FIFOs etc.). This can be used to access any FPGA internal function from the PCI bus.
  - BAR2 configuration must not be changed when any esd drivers (PciAccess, backplane-network-driver) are used.

Please read the bootloader chapter of this manual to get detailed information on how to modify the PCI BARx configuration.

11 Bootloader

11.1 License

The PMC-CPU/440 module uses the opensource bootloader *„Das U-Boot“*. The U-Boot sourcecode is published in terms of the GNU public license (GPL). Please see the appendix of this document for the full license text. You can contact esd for a copy of the full bootloader sourcecode for the PMC-CPU/440.

Esd electronics will take every effort to upload all PMC-CPU/440 board specific changes on the U-Boot sourcecode back to the official repository. The U-Boot project homepage is at http://www.denx.de/wiki/UBoot.

11.2 Configuration and Console Access

Use an USB cable with mini-B connector (PMC440 side) and type A connector (PC side) to connect the PMC-CPU/440 to a PCs USB port. The U-Boot console is accessable via the frontpanel's USB

'CON' device port (mini-B socket). After the first power-on of the PMC module you will be prompted for a driver. You should have received a suitable driver from esd for MS Windows operating systems (Windows 2000 and above). Most Linux distributions bring their own driver for the used on-board FTDI USB-serial converter. When driver installation has been down you have a new virtual serial port (COMx on Windows and typically /dev/ttyUSBx on Linux). Now open a terminal program and point to the virtual COM port of the PMC-CPU/440. The default communication parameters are 115200 baud, 8N1 (8 databits, no parity, 1 stopbit, no hardware handshake).

After the next power-on you will see the bootloader startup messages being output on the serial console. When you see the message 'Press SPACE ....', hit the space key to stop booting and to access the interactive bootloader console. At the prompt you can use an extensive command set to do configuration, debugging or testing tasks. Enter *help* (followed by hitting the RETURN key) to get a full list of all supported commands.

```
U-Boot 1.3.1 (Jan 21 2008 - 11:41:50)

CPU:   AMCC PowerPC 440EPx Rev. A at 533.334 MHz (PLB=133, OPB=66, EBC=66 MHz)
       No Security/Kasumi support
       Bootstrap Option C - Boot ROM Location EBC (16 bits)
       Internal PCI arbiter disabled, PCI async ext clock used
       32 kB I-Cache 32 kB D-Cache
Board: esd GmbH - PMC440, Rev 1.1, non-monarch, PCI=33 MHz
I2C:   ready
DTT:   1 is 35 C
DTT:   2 is 43 C
DRAM:  256 MB
FLASH:  4 MB
NAND:  256 MiB
In:    serial
Out:   serial
Err:   serial
USB:   Host
Net:   ppc_4xx_eth0, ppc_4xx_eth1
POST cache PASSED
POST i2c PASSED
POST cpu PASSED
POST fpu PASSED
POST ethernet PASSED
POST spr PASSED
PARAM: @0fc00000
Press SPACE to abort autoboot in 3 seconds
=>
```

11.3 Default Bootloader Environment

U-Boot uses so called environment variables to store any configuration. The full set of variables can be printed to the console by executing the *printenv* command. Variables can be modified through the *setenv <name> [<value>]* command and finally modifications can be stored in a non-volatile memory through *saveenv*.

This is the factory default environment setup for the PMC-CPU/440 (only the most important variables are printed here):

```
=> printenv
...
baudrate=115200
loads_echo=1
update=protect off FFFA0000 FFFFFFFF;era FFFA0000 FFFFFFFF;cp.b 200000 FFFA0000 60000
serial#=PMC440_GA0013
hostname=pmc440_GA0013
ethaddr=00:02:27:83:40:18
eth1addr=00:02:27:83:40:19
bd_type=pmc440
bootdelay=3
```

```
preboot=painit
pram=4096
load=tftp 200000 /tftpboot/pmc440/u-boot.bin
ethrotate=no
ethact=ppc_4xx_eth0
fpga=tftp 100000 /tftpboot/pmc440/pmc440_fpga.bit;fpga loadb 0 100000
bootcmd=run fpga
mem=258048k
ver=U-Boot 1.3.1 (Jan 21 2008 - 11:41:50)
gatewayip=10.0.0.79
netmask=255.255.0.0
ipaddr=10.0.18.113
serverip=10.0.0.190
...
Environment size: 1507/4092 bytes
=>
```

11.4 Flash Update

The bootloader resides at the top of the PMC440 onboard NOR flash memory. Assuming a binary size of 384kByte (0x60000) the bootloader image must be programmed into the flash memory starting at 0xfffa0000. Please check that you are using the correct bootloader image that suits to your board!

The bootloader update process is very delicate. Any mistake may result in a board that is not usable anymore and which must be shipped back to be reprogrammed by esd using a JTAG debugger.

Here are step-by-step instructions for bootloader update at the serial console. The tftp command requires a correct U-Boot network configuration. That means the the bootloader environment variables *ipaddr, netmask, gatewayip* and *serverip* must be setup according to you network.

When any of the following step fail, do not proceed!

1. Upload the new bootloader binary image into RAM (e.g. at RAM address 0x100000). The easiest way is to load the images from a TFTP-server.

```
=> tftp 100000 /tftpboot/pmc440/u-boot.bin
```

2. Remove flash write protection:

```
=> protect off fffa0000 ffffffff
```

3. Erase the current bootloader. ***Attention:*** do not switch of the board from now on until a new bootloader has been written into the flash!

```
=> erase fffa0000 ffffffff
```

4. Copy the new bootloader into flash:

```
=> cp.b 100000 fffa0000 60000
```

   This command copies 0x60000 bytes starting at RAM location 0x100000 into flash memory starting at 0xfffa0000.

5. Enable flash write protection for bootloader address range:

```
=> protect on fffa0000 ffffffff
```

6. When all the above commands succeeded reset the board.

```
=> reset
```

To simplify the bootloader update process two environment variables already contain the necessary commands. These variables can be executed using the *run* command:

1. Load binary image from a TFTP-server:

```
=> run load
```

2. When the previous command succeeded: unprotect, erase, flash and reset:

```
=> run update
```

Attention: Please doublecheck the content of the *load* and *update* variable before using them. This will prevent you from running into trouble.

11.5 BSP Commands

The following U-Boot BSP commands have been specially added to support the PMC-CPU/440 functions. Type *help <command>* at the bootloader prompt to get a short command reference.

11.5.1 irigb - Get / Set IRIG-B time

The *irigb* command will display the current IRIG-B time (IRIG-B receiver + local IRIG-B generators time). This command will change in the future, because there will only be a single IRIG-B time.

This command requires a booted FPGA with IRIG-B core.

11.5.2 inta – Assert / Deassert PCI interrupt line on PMC440

This command is only supported in non-monarch configuration. It can be used to manually assert the PCI-INTA# interrupt line for testing purposes.

11.5.3 sbe – configure CPU strapping

The *sbe* command is used to write CPU strapping configuration into an on-board EEPROM. This configuration is automatically read by the CPU on reset when the JP2 jumper (see chapter 4) is not installed. When the JP2 jumper is installed the PMC-CPU/440 modules is configured from compiled-in defaults. So installing the JP2 jumper might be useful to bring up the module after incorrect data has been written to the strapping EEPROM.

Use the help (or '?') command to get a short command reference:

```
=> ? sbe
sbe <cpufreq:400|533|667> [<console-uart:0|1> [<bringup delay (0..20s)>]]
```

sbe takes up to 3 arguments. The first argument is the desired CPU clock frequency. Values of 400, 533 and 667MHz are supported.

Attention: Do not configure the module for 667MHz CPU frequncy when no 667MHz-CPU type is installed.

The second argument configured the bootloader console port. A value of '0' uses the CPU's first serial port for the U-Boot console. This port is available as the USB 'CON' port on the frontpanel. A value of '1' configures the CPU's seconds serial port as bootloader console. This port is available on the PMC-P4 connector. The latter is also the default configuration with jumper JP2 installed.

The last argument is used to setup a bringup delay in units of seconds. Following a board reset the first bootloader message on the serial console is delayed by this value. This is especially helpful when using the USB 'CON' console: on most PC operating systems the serial port device of the PMC-CPU/440 will appear a short time after the device is powered on. This means that a terminal program on the PC cannot open the port before the PMC-CPU/440 module is powered on. With a suitable bringup delay the user has enough time to open the serial port from the terminal program without losing any startup messages.

Example (setup 533MHz cpu frequency, U-Boot console is available via USB 'CON' connector, 5s bringup delay):

```
=> sbe 533 0 5
Bootstrapping for 533MHz
Writing boot EEPROM ...
done (dump via 'i2c md 52 0.1 14')
=>
```

Attention: Do not use this command until you know what you are doing!

11.5.4 fifo – Control Hardware FIFO Module

The *fifo* command is used to demonstrate and debug the FPGA internal hardware FIFOs. The full functionality of the FIFO module is described in a separate chapter.

```
=> ? fifo
fifo wait
fifo read
fifo write fifo(0..3) data [cnt=1]
fifo write address(>=4) data [cnt=1]
  - without arguments: print all fifo's status
  - with 'wait' argument: interrupt driven read from all fifos
  - with 'read' argument: read current contents from all fifos
  - with 'write' argument: write 'data' 'cnt' times to 'fifo' or
'address'
```

Without any arguments the *fifo* command will output a short summary about the four FIFOs state:

```
=> fifo
```

```
fifo level status
_____________________________
 0       0   EMPTY
 1       0   EMPTY
 2       0   EMPTY
 3       0   EMPTY
```

The command *fifo read* will dump all FIFOs' content until the FIFOs are empty. The command *fifo wait* will do the same but using an interrupt. The *fifo wait* will dump FIFO data forever until it is interrupted by pressing CTRL-C.

Finally the *fifo write* command can be used to write data to a FIFO's data port. The FIFO data port can be addressed through giving its physical address or its number. The first way can be used to access a FIFO data port over the PCI bus while the 2nd method only works for a local FIFO access:

```
=> fifo write 84000080 12345678 3
writing 3 x 12345678 to fifo port at address 84000080
=> fifo write 2 aa5555aa 2
writing 2 x aa5555aa to fifo 2
=> fifo
fifo level status
_____________________________
 0       0   EMPTY
 1       0   EMPTY
 2       2   NOT EMPTY
 3       0   EMPTY
=>
```

11.5.5 loadpci – Start PCI firmware loading

The loadpci command can be used on a pci target (adapter) platform to boot software images that are copied to RAM from a PCI master (e.g. system CPU) via the PCI bus („firmware download“). It also provides a simple mechanism to remote execute bootloader commands on the pci target that are issued by a PCI master. In most setups the loadpci command is appended to the bootcmd variable.

```
=> printenv bootcmd
bootcmd=run vxworksargs\;loadpci
```

This is how loadpci works:

1. The command cleared the first 0x20 bytes beginning at the target's local PCI BAR1 base address (=PTM1LA).
2. 0xffffffff is written at PTM1LA.
3. The loadpci command start to check the data at PTM1LA until it differs from 0xffffffff or CTRL-C is pressed. This check is done every millisecond. During this polling state you will see the rolling bar in the bootloader console.
4. When the content at PTM1LA changes, the lower 12 bits are taken as a command token while the complete value with masked lower 12 bits is added to PTM1LA address and finally taken as address argument for further processing (so the address parameter is always 4k aligned). (For PMC405 boards PTM1LA is typically 0x00000000. For PMC440 boards PTM1LA is typically 0x0c000000 (192MB).
5. The 12 bit command token decides what U-Boot does next and how the address parameter is used. Unsupported commands lead to termination of the loadpci command with no action.
   - Command "0" bootm: The U-Boot loadaddr variable is set to the address parameter value. Then the bootm command is issued.
   - Command "1" autoscr: The autoscr command is executed with the address parameter passed as script address.

- Command "2" run command: The command string at the address pointed by the address parameter is executed as if it was entered at the bootloader prompt. The command string must be terminated by a '\0'.

See PCIAccess Driver manual for more details about the 'loadpci' command.

11.5.6 fpga command

The *fpga* command is used to boot the on-board Xilinx Spartan 3E FPGA (XC3S1200E):

```
=> tftp 100000 /tftpboot/pmc440/pmc440_fpga.bit
ENET Speed is 1000 Mbps - FULL duplex connection (EMAC0)
Using ppc_4xx_eth0 device
TFTP from server 10.0.0.190; our IP address is 10.0.111.129
Filename '/tftpboot/pmc440/pmc440_fpga.bit'.
Load address: 0x100000
Loading: ################################
done
=> fpga loadb 0 100000
Bytes transferred = 480225 (753e1 hex)
  design filename = "pmc440_r5.ncd"
  part number = "3s1200eft256"
  date = "2007/11/20"
  time = "17: 7:41"
  bytes in bitstream = 480148
=>
```

Step-by-step instructions to write an alternative bitstream into flash memory and load it into the FPGA after reset:

1. Upload the bitstream into RAM (e.g. at RAM address 0x100000). The easiest way is to load the images from a TFTP-server:

```
=> tftp 100000 /tftpboot/pmc440/pmc440_fpga.bit
```

2. Remove flash write protection:

```
=> protect off fff00000 fff7ffff
```

3. Erase the current flash content:

```
=> erase fff00000 fff7ffff
```

4. Copy the new bootloader into flash:

```
=> cp.b 100000 fff00000 80000
```

   This command copies 0x80000 bytes starting at RAM location 0x100000 into flash memory starting at 0xfff00000

5. Enable protection of the flash area:

```
=> protect on fff00000 fff7ffff
```

6. Add the booting command to the bootcmd (or preboot) variable and save the environment. Typically you will need to add some more commands to the bootcmd variable (e.g. to start your operating system=.

```
=> setenv fpga fpga loadb 0 fff00000
=> setenv bootcmd run fpga
=> saveenv
=>
```

7. Reset the board:

```
=> reset
```

11.5.7 painit Command

The *painit* command does all the setup in order to use the PCIAccess driver with a PMC440. *painit* requires a correctly setup *pram* variable (see later chapter). *painit* is typically called by the *preboot* command:

```
=> setenv preboot painit
=> saveenv
```

The 'painit' command presets the reserved memory (see *pram* variable) in the following way:

1. The reserved memory is zero'd.
2. The bootloader's representation of the bootloader environment is copied at the beginning of the reserved memory.
3. The environment size (e.g. 0x00000800) is written to the last 32bit memory location (e.g. 0x3fffffc).
4. The reserved memory size (see 'pram') in units of bytes is written before the previous value (e.g. 0xffffff8).
5. A CRC32 checksum is calculated over the two 32bit value and written at the third last memory location (e.g. 0xffffff4).
6. A 32bit zero (0x00000000) is written to the fourth last memory location (e.g. 0xffffff0).

Attention: The behavior of painit will change with the next bootloader version because of an CPU errata.

11.5.8 USB

This chapter will be added later!

11.5.9 resetout Command

The PMC440 never asserts PMC RESETOUT# automatically because this would lead to race conditions during power-up or reset cycles. But it is possible to assert RESETOUT# manually using the resetout command.

Assert RESETOUT#:

```
=> resetout 0
```

Attention: Never add this command to any bootloader variable that is automatically executed after power-on (e.g. *preboot* or *bootcmd*). This would lead to an uninteruptible reset loop.

11.6 Special Environment Variables

11.6.1 pcidelay Variable

The booloader variable *pcidelay* can be used to delay PCI enumeration through the bootloader. When *pcidelay* is not set, the bootloader on a PMC440 in monarch mode starts PCI enumeration as soon as possible (EREADY is still checked). When *pcidelay* is set to an amount of time in milliseconds of seconds, the bootloader waits for this period, checks EREADY and finally starts PCI enumeration.

Example:

```
=> setenv pcidelay 2000
=> saveenv
```

11.6.2 ptm1la, ptm1ms, ptm2la and ptm2ms Variables

This set of bootloader environment variables can be used to overwrite the default PCI memory resources that are requested by the PMC module. Each PCI address space of the PPC440EPx's PCI bridge is configured by a set of variable ptmXla and pciXms. X stands for the two address spaces 1 and 2.

- The ptmXla determine the 440EPx local base address for the corresponding PCI address translation.
- PtmXms setup the address window size through a size mask (~(size – 1)). The LSB enables the address space.

When the variables are not set these are the defaults:

- ptm1la = 0, ptm1ms= ~(installed_mem_size – 1) | 1
  - This setup enables BAR1 to map the complete SDRAM
- ptm2la = 0xef000000, ptm2ms = 0xff000001
  - This setup enabled BAR2 to map the 440EPx's internal peripherals (e.g. GPIO registers) and the FPGA internal registers.

See the 440EPx usermanual for more details.

11.6.3 pram Variable

The *pram* variable is used to reserve RAM that is not used by the bootloader. When *pram* is set, the variable *mem* is automatically set to the amount of available RAM (total RAM – pram). The reserved RAM is used by the esd PCIAccess driver.

Pram must be set to the amount of reserved RAM in KiB.

```
=> setenv pram 4096
=> saveenv
```

12 FPGA

The PMC-CPU/440 comes with a Xilinx Spartan 3E FPGA (part#: XC3S1200E-4FTG256C) installed. The FPGA provides several functions that are described in this chapter.

12.1 Functional Blocks

- IRIG-B
  - decoding of B100 timecode format
  - input source selection
    - P4 rear differential input (default)
    - P4 rear TTL input
  - local time code generation
- Multiple CAN IP cores (default: 2, max: 8)
- (CAN-) timestamping unit
  - local time (64 bit counter) with 10us resolution
  - timestamp latching for CAN
  - free running counter, sync'ed to IRIG-B signal (digital PLL) or external clock
    - default: external clock from Pn4
  - resetable via register access or IRIG-B control function flag
  - 100kHz CLOCK_OUT + RESET_OUT generation on Pn4 for simple syncronisation of external peripherals
- Pn4 + frontpanel GPIO configuration
  - usage as GPIO
  - alternative functions (IRIG-B, timestamp syncing signals)
- Hardware FIFOs
- Custom module for customized extensions of the PMC-CPU/440

12.2 FPGA Registers

See chapter 8 for local FPGA base addresses. All registers are 32bit wide. Bit 31 is MSB, bit 0 is LSB.

<table>
<tr><th>Register</th><th>Offset</th><th>Mode</th><th>Function</th></tr>
<tr><td>CTRL(A)</td><td>0x0000</td><td>R/W</td><td>Mode / Control register
- Pn4 IO configuration (GPIO/alternative functions)
(see description below)</td></tr>
<tr><td>STATUS</td><td>0x0004</td><td>R</td><td>Status register
(see description below)</td></tr>
<tr><td>CTRLB</td><td>0x0008</td><td>R/W</td><td>Secondary mode/control register
- front panel LED control
- LED behavior
(see description below)</td></tr>
<tr><td></td><td>0x000c</td><td></td><td>reserved</td></tr>
<tr><td>TSH</td><td>0x0010</td><td>R</td><td rowspan="2">64 bit local timestamp counter
The clock source and frequency can be setup by the TSCTRL register.</td></tr>
<tr><td>TSL</td><td>0x0014</td><td>R</td></tr>
<tr><td>TSCTRL</td><td>0x0018</td><td>R/W</td><td>Timestamp unit control register</td></tr>
<tr><td></td><td>...</td><td></td><td>reserved</td></tr>
<tr><td>TSL0H</td><td>0x0020</td><td>R</td><td rowspan="2">64 bit local timestamp latch for CAN0.
TSH+TSL is latched into these registers when a hardware interrupt from the CAN controller 0 occurs. The register stays unchanged until the next falling edge of the interrupt line.</td></tr>
<tr><td>TSL0L</td><td>0x0024</td><td>R</td></tr>
<tr><td>TSL1H</td><td>0x0028</td><td>R</td><td rowspan="2">64 bit local timestamp latch for CAN1.
TSH+TSL is latched into these register when a hardware interrupt from the CAN controller 1 occurs. The register stays unchanged until the next falling edge of the interrupt line.</td></tr>
<tr><td>TSL1L</td><td>0x002c</td><td>R</td></tr>
<tr><td></td><td>...</td><td></td><td>reserved</td></tr>
<tr><td>IRIG_TIME</td><td>0x0040</td><td>R/W</td><td>IRIG-B time that is used by transmitter. The time can be set by a write to this register.
<table>
<tr><th>Bits</th><th>N</th><th>Function</th></tr>
<tr><td>(MSB) 31...30</td><td>2</td><td>unused</td></tr>
<tr><td>29...20</td><td>10</td><td>Day (2.5 bcd digits: 0..366)</td></tr>
<tr><td>19...14</td><td>6</td><td>Hour (1.5 bcd digits: 0..23)</td></tr>
<tr><td>13...7</td><td>7</td><td>Minute (2.5 bcd digits: 0..59)</td></tr>
<tr><td>6...0 (LSB)</td><td>7</td><td>Second (2.5 bcd digits: 0..59)</td></tr>
</table></td></tr>
<tr><td>IRIG_TOD</td><td>0x0044</td><td>R</td><td>IRIG-B time that is used by transmitter. The SBS field is calculated from the content of the IRIG_TIME register. This register is read-only.
<table>
<tr><th>Bits</th><th>N</th><th>Function</th></tr>
<tr><td>(MSB) 31..17</td><td>15</td><td>unused</td></tr>
<tr><td>16..0</td><td>17</td><td>SBS (straight binary seconds) (0..86399)</td></tr>
</table></td></tr>
<tr><td>IRIG_CF</td><td>0x0048</td><td>R</td><td>IRIG-B control function flags that are inserted into the transmitted frame.</td></tr>
</table>

Register	Offset	Mode	Function
			<table><tr><th>Bits</th><th>N</th><th>Function</th></tr><tr><td>(MSB) 31..27</td><td>5</td><td>unused</td></tr><tr><td>26..0 (LSB)</td><td>27</td><td>CF (control functions) LSB (bit0) corresponds to CF0. CF0 ist the first bit of the control functions filed in the IRIG-B frame.</td></tr></table>
IRIG_CTRL	0x004c	R/W	IRIG-B modul configuration: Bit0: 1=receive framing error
IRIG_RX_TIME	0x0050	R	Received IRIG-B time. Bitfields in this register are identical to IRIG_TIME register.
IRIG_RX_TOD	0x0054	R	Received IRIG-B time. Bitfields in this register are identical to IRIG_TOD register.
IRIG_RX_CF	0x0058	R	Received IRIG-B time. Bitfields in this register are identical to IRIG_CF register.
	...		reserved
HOSTCTRL	0x0060	R/W	Host control register. This register can be used to request an interrupt service by the PPC440. This is an alternative way than a write access to the PCI_COMMAND register for an other PCI device (e.g. host CPU) to trigger an interrupt. The implementation of PCI configuration cycles, as needed for access to the PCI_COMMAND register, is very slow and time consuming on some operating systems (e.g. MS Windows with RTX extension). The HOSTCTRL register also provides interrupt masking capabilities for the FIFO module and the optional FPGA custom module.
	...		reserved
DDFSCTRL	0x0070	R/W	DDFS control register / ADPLL phase error (see detailed description below)
DDFSINC	0x0074	R	DDFS increment value. (see detailed description below)
	...		reserved
FIFO0_DATA	0x0080	R/W	FIFO 0 data port
FIFO0_CTRL	0x0084	R/W	FIFO 0 control/status register
FIFO1_DATA	0x0088	R/W	FIFO 1 data port
FIFO1_CTRL	0x008c	R/W	FIFO 1 control/status register
FIFO2_DATA	0x0090	R/W	FIFO 2 data port
FIFO2_CTRL	0x0094	R/W	FIFO 2 control/status register
FIFO3_DATA	0x0098	R/W	FIFO 3 data port
FIFO3_CTRL	0x009c	R/W	FIFO 3 control/status register

12.3 Register Description

12.3.1 CTRL Register (0x0000)

Bits	Name	Function
31	RESET_EN	When set to '1' the RS485 RESET signal driver is enabled. The RS485 driver circuit is located on the PMC405-PIM module. When enabled the PMC440 is the source of the RESET signal. When the PMC440 is not used with the PMC405-PIM this flag can be used to control the state of the TTL signal RESET_R_EN on Pn4 (pin#62). Default after reset: '0'
30	CLOCK_EN	When set to '1' the RS485 CLOCK signal driver is enabled. The RS485 driver circuit is located on the PMC405-PIM module. When enabled the PMC440 is the source of the CLOCK signal. When the PMC440 is not used with the PMC405-PIM this flag can be used to control the state of the TTL signal CLOCK_R_EN on Pn4 (pin#60). Default after reset: '0'
29	CLKRST_EN_CSTM	When set to '1' the RESET_EN and CLOCK_EN signals on Pn4 (see above) are connected to the FPGA's custom module. Default after reset: '0'
28	IRIGB_R_EN	When set to '1' the RS485 output driver for IRIG-B transmission on Pn4 on the PMC440 is enabled and IRIG_B_R_OUT_P/_M become a differential output. Default after reset: '0'
27	IRIGB_R_EN_CSTM	When set to '1' the IRIGB_R_EN signal (driver enable for RS485 transceiver) is connected to the FPGA's custom module. Default after reset: '0'
26	reserved	Default after reset: '0'
25	reserved	Default after reset: '0'
24..22	CLOCK_MODE[2..0]	'000': CLOCK_OUT signal is driven low '001': CLOCK_OUT signal is driven high '010': 100kHz, 50% duty cycle clock with IRIG-B sync option '111': CLOCK_OUT is connected to the FPGA's custom module. Default after reset: '000'

Bits	Name	Function
21..19	RESET_MODE[2..0]	'000': RESET_OUT signal is driven low '001': RESET_OUT signal is driven high '010': RESET_OUT pulse is triggered by manual timestampcounter reset (see timestamp unit) '011': RESET_OUT pulse is triggered by IRIG-B receiver unit (CF). '100': esd internal test mode – do not use '111': RESET_OUT is connected to the FPGA's custom module. Default after reset: '000' The automatically timed reset pulse has about 6 us pulse width.
18..16	IRIGB_SRC[2..0]	IRIG-B input select for IRIG-B module: '000': IRIG_B_R_P/M (Pn4 diff. input) '001': IRIG_B_R_IN (Pn4 TTL input) '010': reserved '011': reserved '1xx': reserved for future use Default after reset: '000'
15..14	IRIGB_R_OUT_MODE[1..0]	configuration of IRIG-B TTL output on Pn4 '00': signal is driven low '01': signal is driven high '10': IRIG-B output '11': signal is connected to the FPGA's custom module. Default after reset: '00'
13..12	IRIGB_R_OUTPM_MODE[1..0]	configuration of IRIG-B differential output on Pn4 '00': signal is driven low '01': signal is driven high '10': IRIG-B output '11': signal is connected to the FPGA's custom module. Default after reset:1 '00'
11..9	reserved	Default after reset: '000'
8	HOST_IE	Host control interrupt enable. See HOSTCTRL register.
7..4	reserved	Default after (FPGA-) reset: '0000'
3..2	reserved	Default after (FPGA-) reset: '00'
1	reserved	Default after (FPGA-) reset: '0'
0	reserved	Default after (FPGA-) reset: '0'

12.3.2 STATUS Register (0x0004)

The logic state readback functionality is independant from any alternative function.

Bits	Name	Function
(MSB) 31..24	VERSION	FPGA version (current version is 0x07)
23..20	HW_REV	PMC440 hardware revision (1.x, 0 <= x <= 15) This bit field is updated by the bootloader after FPGA booting (see chapter 5 on how to query the hardware revision).
19	ETH1_LED2	Ethernet phy 1 – LED2 output readback (testing only)
18	ETH1_LED1	Ethernet phy 1 – LED1 output readback (testing only)
17	ETH0_LED2	Ethernet phy 0 – LED2 output readback (testing only)
16	ETH0_LED1	Ethernet phy 0 – LED1 output readback (testing only)
15..11	reserved	unused (read '0')
12	CANIP_ISF	CAN IP core interrupt status flag. This bit indicates a pending interrupt from any CAN controller.
11	CSTM1_ISF	Custom module interrupt status flag. This bit indicates a pending interrupt from the FPGA's custom module („nIRQ1“, „nIRQ2“).
10	CSTM0_ISF	Custom module interrupt status flag. This bit indicates a pending interrupt from the FPGA's custom module („nIRQ0“).
9	FIFO_ISF	FIFO module interrupt status flag. This bit indicates a pending interrupt from the FIFO module.
8	HOST_ISF	Host interrupt status flag. This bit indicates a pending host interrupt.
7 .. 6	reserved	unused (read '0')
5	IRIGB_R_IN	logic state of Pn4 TTL input signal
4	IRIGB_R_IN_PM	logic state of Pn4 RS485 input signal
3	reserved	unused (read '0')
2	reserved	unused (read '0')
1	CLOCK_IN	logic state of Pn4 CLOCK input signal
0	RESET_IN	logic state of Pn4 RESET input signal

12.3.3 CTRLB Register (0x0008)

Bits	Name	Function
(MSB) 31..10	reserved	unused -reads always 0
9	LEDB_MODE	When set to '1' LEDB (frontpanel) is controlled by CTRLB[LEDB]. When set to '0' LEDB is pulsed by an interrupt coming from an odd (1,3,...) CAN IP core or by nIRQ2 from the FPGA's custom module. Default after reset: '1'

Bits	Name	Function
8	LEDA_MODE	When set to '1' LEDA (frontpanel) is controlled by CTRLB[LEDA]. When set to '0' LEDA is pulsed by an interrupt coming from an even (0,2,...) CAN IP core or by nIRQ1 from the FPGA's custom module. Default after reset: '1'
7..2	reserved	unused -reads always 0
1	LEDB	Controls the state of LEDB on the PMC's frontpanel when CTRLB[LEDB_MODE] bit is set to '1'. Default after reset: '0'
0	LEDA	Controls the state of LEDA on the PMC's frontpanel when CTRLB[LEDA_MODE] bit is set to '1'. Default after reset: '0'

12.3.4 TSCTRL - Timestamp unit control register (0x0018)

Bits	Name	Function
(MSB) 31..5	reserved	unused -reads always 0
5	TS_CFRSTEN	Enable timestamp counter reset via IRIG-B control flags
4	TS_MRST	Timestamp manual reset -write '1' to reset timestamp counter -reads always '0'
3	TS_XRSTEN	Timestamp reset enable: '1': enable reset of timestamp counter via external RESET signal on Pn4
2..0	TS_SRC[2..0]	Timestampcounter clock source select: '000': FPGA internal 100kHz clock '010': external CLOCK signal on Pn4 (other combinations of the TS_SRC bits are reserved for future use.)

Note: Write 0x0000.0000 to TSCTRL for testing and read the current 64 bit counter value from offset 0x0010 and 0x0014. This uses the FPGA internal 100kHz timestamp clock source.

12.3.5 HOSTCTRL – Host control register (0x0060)

Bits	Name	Function
31 .. 6	RESERVED	Always read '0'
17	PMCRSTOUT_GATE	see PMCRSTOUT_FLAG
16	PMCRSTOUT_FLAG	This bit represents the state of PMC-RESETOUT# signal. After reset this bit is set. Clearing this bit can be used to assert the PMC RESETOUT# signal. The PMCRSTOUT_FLAG can only be modified when the PMCRSTOUT_GATE bit is set during the same write access to the HOSTCTRL register. This makes modification of the PMCRSTOUT_FLAG possible without read-modify-write operations and without taking care of the other bits of the HOSTCTRL register.
15 .. 8	RESERVED	Always read '0'

Bits	Name	Function
7	CSTM1IE_GATE	see CSTM1IE_FLAG
6	CSTM1IE_FLAG	'1': unmask interrupts from the FPGA custom module. '0': mask interrupts from the FIFO module. The CSTM1IE_FLAG can only be modified when the CSTM1IE_GATE bit is set during the same write access to the HOSTCTRL register. This makes modification of the CSTM1IE_FLAG possible without read-modify-write operations and without taking care of the other bits of the HOSTCTRL register.
5	CSTM0IE_GATE	see CSTM0IE_FLAG
4	CSTM0IE_FLAG	'1': unmask interrupts from the FPGA custom module. '0': mask interrupts from the FIFO module. The CSTM0IE_FLAG can only be modified when the CSTM0IE_GATE bit is set during the same write access to the HOSTCTRL register. This makes modification of the CSTM0IE_FLAG possible without read-modify-write operations and without taking care of the other bits of the HOSTCTRL register.
3	FIFOIE_GATE	see FIFOIE_FLAG
2	FIFOIE_FLAG	'1': unmask interrupts from the FIFO module. FIFO interrupts must also be enabled for each FIFO in the corresponding FIFOx_CTRL register. '0': mask interrupts from the FIFO module. The FIFOIE_FLAG can only be modified when the FIFOIE_GATE bit is set during the same write access to the HOSTCTRL register. This makes modification of the FIFOIE_FLAG possible without read-modify-write operations.
1	HCINT_GATE	see HCINT_FLAG
0	HCINT_FLAG	'1': assert host interrupt (host interrupts must be enabled via HOST_IE bit in the CTRL register) '0': deassert host interrupt The HCINT_FLAG can only be modified when the HCINT_GATE bit is set during the same write access to the HOSTCTRL register. This makes modification of the HCINT_FLAG possible without read-modify-write operations.

12.3.6 DDFSCTRL/DDFSINC – Clock generator registers (0x0070, 0x0074)

The FPGA uses a DDFS (direct frequency synthesis clock generator to synthesize a 100kHz reference clock. This clock can be output on Pn4 and/or directly passed to the timestamp unit. The reference clock can optionally be synchronized through an IRIG-B timecode signal and the frequency is controlled by an FPGA internal ADPLL.

The current reference clock frequency is calculated as

$$f_{100kHz} = 66.6668\ MHz * DDFSINC / 2^{32}.$$

Base value with IRIG-B sync regulator disabled is 0x00624e00. When the IRIG-B sync regulator is active the DDFSINC register is updated every IRIG-B bit time.

The DDFS unit is can be controlled through the DDFSCTRL register:

Bits	Name	Function
31 .. 3	RESERVED	Always read '0'
2	RESTART	'1' : reference clock is stopped an regulator parameter are reset. '0': reference clock is enabled. When REGULATE flag is set, the oscillator starts with the next IRIG-B bit. Default after (FPGA-) reset is: '0'
1	REGULATE	'1': reference clock is synchronized to IRIG-B input source. '0': reference clock is free running Default after (FPGA-) reset is: '0'
0	STOP	'1': reference clock is stopped and regulator parameter a left untouched. '0': reference clock is enabled. When REGULATE flag is set, the oscillator starts with the next IRIG-B bit. Default after (FPGA-) reset is: '0'

12.3.7 FIFO<0...3>_DATA

32 bit wide FIFO data port. Data producers write to this register. Consumers read from this register. Only 32 bit access to this register is supported.

12.3.8 FIFO<0...3>_CTRL

FIFO control register. Each register controls the behavior of a single FIFO.

Bits	Name	Access	Function
15	IE	R/W	Enable interrupts from this FIFO. An interrupt is asserted when the FIFO contains at least one entry (EMPTY=0). The interrupt is reset when all data is read from the FIFO. Clearing the IE bit only masks the interrupt.
11 ... 14	RESERVED	R	Always read '0'
10	OVERFLOW	R/W	This flag is set when a write access to a full FIFO occured, which means that data got lost. Writing a '0' resets the OVERFLOW flag.
9	EMPTY	R	FIFO empty. The FIFO does not contain any valid data.
8	FULL	R	FIFO full. The FIFO contains exactly 256 valid entries that are ready to be read.
7...0	LEVEL	R	Filling level of FIFO (0=empty). Each write access to the data port increases LEVEL while a read access decreases LEVEL. LEVEL=0 means the FIFO is empty (EMPTY=1). A full fifo is indicated by LEVEL=0x00 and the fifo full flag set to '1'. This means bits 8..0 can be interpreted as the total fifo level from

Bits	Name	Access	Function
			0x000 to 0x100.

12.4 FPGA Interrupts

FPGA-Interrupt	PPC440 UIC/Int#	FPGA Interrupt Source	Mask Register[bit]	Status Register[bit]
IRQ0	1 / 28	FIFO Module	HOSTCTRL[FIFO_IE_FLAG]	STATUS[FIFO_ISF]
		Host Interrupt	HOSTCTRL[HOST_IE_FLAG]	STATUS[HOST_ISF]
IRQ1	1 / 30	Custom Module „nIRQ0“	HOSTCTRL[CSTM0IE_FLAG]	STATUS[CSTM0_ISF]
IRQ2	2 / 3	CAN IP Cores	HOSTCTRL[CANIPIE_FLAG]	STATUS[CANIP_ISF]
		Custom Module „nIRQ1“, „nIRQ2“	HOSTCTRL[CSTM1IE_FLAG]	STATUS[CSTM1_ISF]

12.5 Using the FIFO module

The PMC440 provides four hardware FIFOs that are implemented in the on-board FPGA. Each FIFO provides a 32 bit data port and has a capacity of 256 entries. The FIFOs share a common external interrupt to the PPC440 CPU. Each FIFO is represented by a set of two 32 bit registers. The registers are accessible from the PPC440 and from the PCI bus. This allows other PCI devices to write data into the FIFOs data ports. The following sample code demonstrates the FIFO module usage. The extract is taken from the PMC405V2 U-Boot sourcecode. The code will be adapted to the PMC440 soon.

```
/* FPGA interface */
#define FPGA_OUT32(p,v) out32((int)(p), (v))
#define FPGA_IN32(p) in32((int)(p))
#define FPGA_SETBITS(p,v) out32((int)(p), in32((int)(p)) | (v))
#define FPGA_CLRBITS(p,v) out32((int)(p), in32((int)(p)) & ~(v))

struct pmc405v2_fifo_s {
      u32 data;
      u32 ctrl;
};

/* bits in fifo ctrl register */
#define FIFO_IE              (1 << 15)
#define FIFO_OVERFLOW        (1 << 10)
#define FIFO_EMPTY           (1 <<  9)
#define FIFO_FULL            (1 <<  8)
#define FIFO_LEVEL_MASK 0x000000ff

#define FIFO_COUNT      4

struct pmc405v2_fpga_s {
      u32 ctrl;
      u32 status;
      u32 pad1[0x40 / sizeof(u32) - 2];
      u32 irig_time;                  /* offset: 0x0040 */
      u32 irig_tod;
      u32 irig_cf;
      u32 pad2;
      u32 irig_rx_time;               /* offset: 0x0050 */
      u32 pad3[3];
```

```
	u32 hostctrl;                   /* offset: 0x0060 */
	u32 pad4[0x20 / sizeof(u32) - 1];
	struct pmc405v2_fifo_s fifo[FIFO_COUNT]; /* 0x0080..0x009f */
};

typedef struct pmc405v2_fpga_s pmc405v2_fpga_t;

/* status register */
#define STATUS_FIFO_ISF      (1 <<  9)

/* hostctrl register */
#define HOSTCTRL_FIFOIE_GATE (1 <<  3)
#define HOSTCTRL_FIFOIE_FLAG (1 <<  2)

/* FPGA to PPC interrupt */
#define FPGA_IRQ 31

int got_fifoirq;

/* FIFO module interrupt handler */
int fpga_interrupt(pmc405v2_fpga_t *fpga)
{
	int rc = -1; /* not for us */
	u32 status = FPGA_IN32(&fpga->status);

	/* check for interrupt from fifo module */
	if (status & STATUS_FIFO_ISF) {
		/* disable/mask this int source */
		FPGA_OUT32(&fpga->hostctrl, HOSTCTRL_FIFOIE_GATE);
		rc = 0;
		got_fifoirq = 1; /* trigger backend */
	}
	return rc;
}


/* pretty print content of fifo 'f' */
void dump_fifo(pmc405v2_fpga_t *fpga, int f, int *n)
{
	u32 ctrl;

	while(!((ctrl = FPGA_IN32(&fpga->fifo[f].ctrl)) & FIFO_EMPTY)) {
		printf("%5d  %d    %3d  %08x",
			(*n)++, f, ctrl & (FIFO_LEVEL_MASK | FIFO_FULL),
			FPGA_IN32(&fpga->fifo[f].data));
		if (ctrl & FIFO_OVERFLOW) {
			printf(" OVERFLOW\n");
			FPGA_CLRBITS(&fpga->fifo[f].ctrl, FIFO_OVERFLOW);
		} else {
			printf("\n");
		}
	}
}

/* 'fifo' command from PMC405V2 U-Boot sourcecode */
int do_fifo(cmd_tbl_t *cmdtp, int flag, int argc, char *argv[])
{
	pmc405v2_fpga_t *fpga = (pmc405v2_fpga_t *)FPGA_BA;
	int i;
	int n = 0;
	u32 ctrl, data, f;
	char str[] = "\\|/-";
	int abort = 0;
	int count = 0;
	int count2 = 0;

	switch(argc) {
	case 1:
```

```
            /* print all fifos status information */
            printf("fifo level status\n");
            printf("_____________________________\n");
            for (i=0; i<FIFO_COUNT; i++) {
                  ctrl = FPGA_IN32(&fpga->fifo[i].ctrl);
                  printf(" %d    %3d  %s%s%s %s\n",
                        i, ctrl & (FIFO_LEVEL_MASK | FIFO_FULL),
                        ctrl & FIFO_FULL ? "FULL     " : "",
                        ctrl & FIFO_EMPTY ? "EMPTY    " : "",
                        ctrl & (FIFO_FULL|FIFO_EMPTY) ? "" : "NOT EMPTY",
                        ctrl & FIFO_OVERFLOW ? "OVERFLOW" : "");
            }
            break;

      case 2:
            /* completely read out fifo 'n' */
            if (!strcmp(argv[1],"read")) {
                  printf("  #   fifo level data\n");
                  printf("_____________________________\n");

                  for (i=0; i<FIFO_COUNT; i++) {
                        dump_fifo(fpga, i, &n);
                  }
            } else if (!strcmp(argv[1],"wait")) {
                  got_fifoirq = 0;

                  irq_install_handler (FPGA_IRQ,
                                       (interrupt_handler_t *)fpga_interrupt, fpga);

                  printf("  #   fifo level data\n");
                  printf("_____________________________\n");

                  /* enable all fifo interrupts */
                  FPGA_OUT32(&fpga->hostctrl,
                             HOSTCTRL_FIFOIE_GATE | HOSTCTRL_FIFOIE_FLAG);
                  for (i=0; i<FIFO_COUNT; i++) {
                        /* enable interrupts from all fifos */
                        FPGA_SETBITS(&fpga->fifo[i].ctrl, FIFO_IE);
                  }

                  while (1) {
                        /* wait loop */
                        while(!got_fifoirq) {
                              count++;
                              if (!(count % 100)) {
                                    count2++;
                                    putc(0x08); /* backspace */
                                    putc(str[count2 % 4]);
                              }

                              /* Abort if ctrl-c was pressed */
                              if ((abort = ctrlc())) {
                                    puts("\nAbort\n");
                                    break;
                              }
                              udelay(1000);
                        }
                        if (abort) {
                              break;
                        }

                        /* simple fifo backend */
                        if (got_fifoirq) {
                              for (i=0; i<FIFO_COUNT; i++) {
                                    dump_fifo(fpga, i, &n);
                              }
                              got_fifoirq = 0;
                              /* unmask global fifo irq */
                              FPGA_OUT32(&fpga->hostctrl,
```

```
                                        HOSTCTRL_FIFOIE_GATE | HOSTCTRL_FIFOIE_FLAG);
                        }
                }

                /* disable all fifo interrupts */
                FPGA_OUT32(&fpga->hostctrl, HOSTCTRL_FIFOIE_GATE);
                for (i=0; i<FIFO_COUNT; i++) {
                        FPGA_CLRBITS(&fpga->fifo[i].ctrl, FIFO_IE);
                }
                irq_free_handler(FPGA_IRQ);

        } else {
                printf ("Usage:\nfifo %s\n", cmdtp->help);
                return 1;
        }
        break;

    case 4:
    case 5:
        if (!strcmp(argv[1],"write")) {
                /* get fifo number or fifo address */
                f = simple_strtoul(argv[2], NULL, 16);

                /* data paramter */
                data = simple_strtoul(argv[3], NULL, 16);

                /* get optional count parameter */
                n = 1;
                if (argc >= 5) {
                        n = (int)simple_strtoul(argv[4], NULL, 10);
                }

                if (f < FIFO_COUNT) {
                        printf("writing %d x %08x to fifo %d\n",
                                n, data, f);
                        for (i=0; i<n; i++) {
                                FPGA_OUT32(&fpga->fifo[f].data, data);
                        }
                } else {
                        printf("writing %d x %08x to fifo port at address %08x\n",
                                n, data, f);
                        for (i=0; i<n; i++) {
                                out32(f, data);
                        }
                }
        } else {
                printf ("Usage:\nfifo %s\n", cmdtp->help);
                return 1;
        }
        break;

    default:
        printf ("Usage:\nfifo %s\n", cmdtp->help);
        return 1;
    }
    return 0;
}
U_BOOT_CMD(
    fifo,  5,     1,     do_fifo,
    "fifo    - Fifo module operations\n",
    "wait\nfifo read\n"
    "fifo write fifo(0..3) data [cnt=1]\n"
    "fifo write address(>=4) data [cnt=1]\n"
    "  - without arguments: print all fifo's status\n"
    "  - with 'wait' argument: interrupt driven read from all fifos\n"
    "  - with 'read' argument: read current contents from all fifos\n"
    "  - with 'write' argument: write 'data' 'cnt' times to 'fifo' or 'address'\n"
    );
```

12.6 FPGA Custom Module

The FPGA custom module allows users to add customized functionality to the PMC440 onboard FPGA. For extending the PMC440 FPGA esd provides the following files (shipped in a zip archive) that have to be added to an Xilinx ISE design project:

- pmc440_cstm_fpga_r7/pmc440.vhd
  (top level VHDL source for PMC440 FPGA design)
- pmc440_cstm_fpga_r7/custom_module_pmc440.vhd
  (example VHDL sourcecode to custom extension – wrapper for custom modules)
- pmc440_cstm_fpga_r7/simple_io_custom_module_pmc440.vhd
  (example VHDL sourcecode to custom extension – simple GPIO controller)
- pmc440_cstm_fpga_r7/pmc440_custom_module_pimesc.vhd
  (example VHDL sourcecode to custom extension – EtherCAT PIM module)
- pmc440_cstm_fpga_r7/pmc440_r7.ucf
  (constraints file)
- pmc440_cstm_fpga_r7/pmc440_common.ngc
  (esd IP of fixed design parts as described in previous chapter of this document)
- pmc440_cstm_fpga-r7/README_CustomDesign.txt
  (information text file)

After implementing the FPGA design the genererated .bit file can directly be loaded into the PMC440 FPGA by using the bootloader's 'fpga' command (see above).

12.6.1 Custom Module Conventions

Some rules should be followed when implementing customized modules for the PMC440 FPGA:

- Each module must provide a 32 bit module ID at offset 0 in the 32 bit custom address space.
- IDs from 0x0000.0000 to 0x7fff.ffff are reservedfor esd. So non-esd custom modules must have the MSB of the ID field set to '1'.
- Each module must provide a 32 bit module version at offset 0x10 in the 32 bit custom address space.

12.6.2 Sample Custom Module „simple_io“

The PMC-CPU/440 is shipped with a default FPGA image that includes the *simple_io* custom module. This module implements a simple GPIO controller for 48 usable FPGA I/Os that are available on the PMC's P4 connector.

The module is controlled by a couple of 32 bit registers. The base address for these registers is ADDR_CSTM32_BASE = 0xef08.0000 (see chap. 8).

Registername	Offset	Function
CSTM32_ID	0x0000	simple_io module identification (0x0000.0001)
CSTM32_VER	0x0010	simple_io module version (starting with 0x0000.0001)
CSTM32_OR0	0x0020	Output register 0 – each bit in this register controls the output level of a corresponding signal on the PMCs P4: FPGA-IO<31> .. FPGA-IO<0>. In order to control the output state the pin must be

		enabled as output via the TCR0 register (see below). Default: 0x0000.0000
CSTM32_TCR0	0x0024	Tristate control register 0 – each bit in this register controls the direction of a corresponding signal on the PMCs P4: FPGA-IO<31> .. FPGA-IO<0>. A set bit enables the signal as output. Default: 0x0000.0000
CSTM32_IR0	0x0028	Input register 0 – each bit mirrows the actual state of a corresponding signal on the PMCs P4: FPGA-IO<31>..FPGA-IO<0>.
CSTM32_OR1	0x0030	Output register 1 – each bit in this register controls the output level of a corresponding signal on the PMCs P4: FPGA-IO<47> .. FPGA-IO<32>. In order to control the output state the pin must be enabled as output via the TCR0 register (see below). Default: 0x0000.0000
CSTM32_TCR1	0x0034	Tristate control register 0 – each bit in this register controls the direction of a corresponding signal on the PMCs P4: FPGA-IO<47> .. FPGA-IO<32>. A set bit enables the signal as output. Default: 0x0000.0000
CSTM32_IR1	0x0038	Input register 0 – each bit mirrows the actual state of a corresponding signal on the PMCs P4: FPGA-IO<47>..FPGA-IO<32>.

13 Ordering information

Type	Description	Order-no.
PMC-CPU/440	PowerPC PrPMC module	V.2027.02
PMC-CPU/440-667	PowerPC PrPMC module (667 MHz version)	V.2027.03
PMC-CPU/440-ME	Usermanual in english	V.2027.21
PIM-CPU/405 2xCAN	PIM module with 2 CAN physical layers	V.2025.02
PIM-CPU/405-PBOX 2xCAN	PIM module with 2 CAN physical layers and PMC-Box clock/reset distribution logic	V.2025.04
PIM-CPU/440-PBOX-ESC	PIM module with Beckhoff ET1100 EtherCAT slave interface	V.2028.02
PMC-CPU/440-VxW	VxWorks BSP	V.2027.30
PMC-CPU/440-Linux	Linux BSP	V.2027.32
PMC-CPU/440-QNX	QNX BSP	V.2027.33
PMC-CPU/440-OS9	OS/9 BSP	V.2027.34

14 Appendix

14.1 GNU General Public License

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's

source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.

You may charge a fee for the physical act of transferring a copy, and
you may at your option offer warranty protection in exchange for a fee.

  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.

    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.

    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License. (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works. But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.

In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software interchange; or,

    b) Accompany it with a written offer, valid for at least three
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,

    c) Accompany it with the information you received as to the offer
    to distribute corresponding source code. (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for
making modifications to it. For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable. However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.

If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not

compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

```
  11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW.  EXCEPT WHEN
OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES
PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED
OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS
TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU.  SHOULD THE
PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,
REPAIR OR CORRECTION.

  12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED
TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY
YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGES.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License along
    with this program; if not, write to the Free Software Foundation, Inc.,
    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

    Gnomovision version 69, Copyright (C) year name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.
```